سر الحُبْ ..

University of Bahrain
College of Information Technology
Department of Computer Science
First Semester, 2011-2012
ITCS215 (Data Structures)

Test II

Date: 26/12/2011 Time: 11:00 - 12:30

STUDENT NAME	
STUDENT ID #	
SECTION #	

NOTE: THERE ARE SIX **(6) PAGES** IN THIS TEST
ONLY ONE SOLUTION WILL BE CONSIDERED FOR EACH QUESTION

QUESTION #	MARKS		COMMENTS
1	15		
2	10		
3	15		
TOTAL	40		

http://www.uob-bh.com/forum/forum.php

Question 1 [9 + 6 Marks] .. سر الحُبْ

(A) [9 Marks] Write a non-member function called **matchWithArray** that accepts an object ***st*** of type **stackType**, an array ***arr*** of type **Type** and ***count*** (number of elements in the array) as parameters. The function compares the elements of the stack ***st*** with the elements of the array ***arr***. If an element is contained in the stack ***st*** but is not contained in the array ***arr*** then the function deletes that element from the stack. The function should keep the remaining elements in their original relative order in stack ***st***.

Assume that class **stackType** is available for use. Use only common stack operations such as push, pop, top, isEmptyStack, isFullStack, operator= and copy constructor.

NOTE: Stack ***st*** may contain more than one copy of some elements. The order of elements in stack ***st*** may not be same as in array ***arr***.

Function prototype:

```
void matchWithArray(stackType<Type>  &st, Type  arr[ ], int count);
```

Example:

Before function call:

st: 4 **6** 5 8 3 7 **2** 8 1 **9** **2**

arr: 1 3 5 7 4 8

After function call:

st: 4 5 8 3 7 8 1

arr: 1 3 5 7 4 8

Note that 6, 2 and 9 are deleted from **st** as they are not there in **arr**.

http://www.uob-bh.com/forum/forum.php

(B) [6 Marks] Consider the following postfix expression. Use stack to evaluate it and show all the push and pop operations by clearly drawing the stack status. سر الحُبّ ..

5 11 7 + 2 3 1 * + - -

http://www.uob-bh.com/forum/forum.php

Question 2 [10 Marks] .. سر الحُبْ

In a bank, there are two counters. In front of the first counter, there are customers waiting in a queue to be served while the second counter is closed. The second counter opens its window, and **half** of the customers are asked to move to the second counter to be served so that the last customer of the first counter's queue will be the first customer of the second counter's queue.
Write a non member function **splitQueue** using the class **queueType** for the above problem..
Function prototype:
void splitQueue(QueueType<Type>& counter1, QueueType<Type>& counter2) ;

Example:
Before **splitQueue()** call:

9	8	7	6	5	4	3	2	1	**Counter-1's window (Queuefront)**
									Counter-2's window (Queuefront)

After **splitQueue** call:

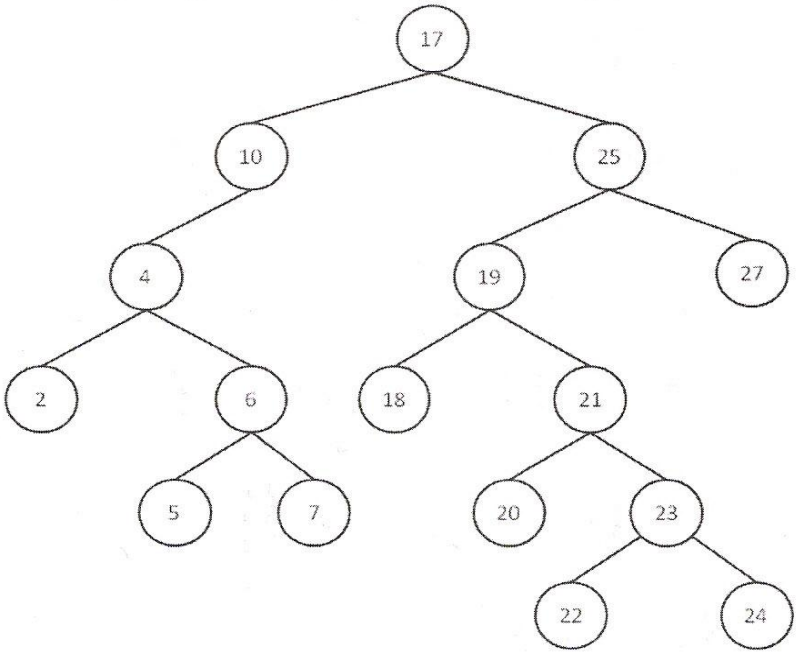
				5	4	3	2	1	**Counter-1's window (Queuefront)**
					6	7	8	9	**Counter-2's window (Queuefront)**

Assume that class **queueType** is available for use. Use only common queue operations such as addQueue, deleteQueue, front, back, isEmptyQueue, isFullQueue, operator= and copy constructor.

http://www.uob-bh.com/forum/forum.php

Question 3 [7 + 8 Marks] .. سر الحُبّ

(A) For the binary tree given below, answer the following questions:



i. **[2 Marks]** What is the **height** of this binary tree?

ii. **[5 Marks]** List the sequence of nodes, if the binary tree is traversed using **post-order traversal.**

http://www.uob-bh.com/forum/forum.php

(B) [8 Marks] Write a recursive private member function called **writToQueue** to be included in class `binaryTreeType`. The function accepts a pointer **p** to a node of a binary tree as a parameter. The function also accepts an object **q** of type **queueType** as parameter. The function copies the **info** of the nodes of the binary tree in queue **q** in **preorder traversal** sequence. If the binary tree is empty, then the queue will also be empty.

This function is called from a public member function **BTWriteToQueue**, given as follows:

```
template<class elemType>
void binaryTreeType<elemType>::BTWriteToQueue(queueType<elemType>& q)
{
      q.initializeQueue( );
      writeToQueue(root, q);
}
```

Function Prototype: .. سر الحُبّ

```
void writeToQueue(nodeType<elemType> *p, queueType<elemType>& q);
```


http://www.uob-bh.com/forum/forum.php